**COSC 436: Object-Oriented Design and Programming**

**In-class Exercise: Factory Method Design Pattern**

**Problem:**

The objective of this exercise is to implement the Factory Method design pattern.

**Tasks:**

In this exercise, **Buttons** play a product role and **Dialogs** act as creators. Different types of dialogs require their own types of elements. We will create a subclass for each **dialog** type and **override** their factory methods. Each **dialog** type will instantiate proper **button** classes. **Base dialog** works with products using their common interface, so its code remains functional after all changes.

1. We have an interface, called **Button**, which defines two methods, **render**() and **onClick**().

A black background with colorful text

Description automatically generated

1. Create an **HtmlButton** class, which implements Button. Provide the implementation for both render() and onClick() methods.

**public** **void** render() {

System.out.println("<button>Test Button</button>");

onClick();

}

**public** **void** onClick() {

System.out.println("Click! Button says - 'Hello World!'");

}

A screen shot of a computer program

Description automatically generated

1. Create a **WindowsButton** class, which also implements Button. Provide the implementation for both render() and onClick() methods.

JPanel panel = **new** JPanel();

JFrame frame = **new** JFrame();

JButton button;

**public** **void** render() {

}

**public** **void** onClick() {

}

}

A screen shot of a computer program

Description automatically generated

1. Create a base creator, called **Dialog**. Have two methods in it.

**public** **void** renderWindow() {

}

**public** **abstract** Button createButton();

A screen shot of a computer program

Description automatically generated

1. Create two concrete creators: **HtmlDialog** and **WindowsDialog**, by providing implementation for the abstract method createButton(). One returns new WindowsButton(), the other returns new HtmlButton().

A screen shot of a computer program

Description automatically generated

A computer screen shot of text

Description automatically generated

1. Use the following client code to test it.

**public** **class** **Client** {

**private** **static** Dialog dialog;

**public** **static** **void** main(**String**[] args) {

configure();

run();

}

*/\*\**

*\* The concrete factory is usually chosen depending on configuration or*

*\* environment options.*

*\*/*

**static** **void** configure() {

**if** (System.getProperty("os.name").equals("Windows 10")) {

dialog = **new** WindowsDialog();

} **else** {

dialog = **new** HtmlDialog();

}

}

*/\*\**

*\* All of the client code should work with factories and products through*

*\* abstract interfaces. This way it does not care which factory it works*

*\* with and what kind of product it returns.*

*\*/*

**static** **void** run() {

dialog.renderWindow();

}

}

A screenshot of a computer

Description automatically generated

A screenshot of a message

Description automatically generated

1. Can you draw a class diagram for these classes?

Upload your code to the Blackboard when you are done.